

Analysis of Railway Track Irregularities with Convolutional Autoencoders and Clustering Algorithms^{*}

Julia Niebling¹[0000-0001-5413-2234], Benjamin Baasch²[0000-0003-1970-3964],
and Anna Kruspe¹[0000-0002-2041-9453]

¹ Deutsches Zentrum für Luft- und Raumfahrt, Institute of Data Science,
Mälzerstraße 3, 07745 Jena, Germany

² Deutsches Zentrum für Luft- und Raumfahrt, Institute of Transportation Systems,
Rutherfordstr. 2, 12489 Berlin, Germany

Abstract. Modern maintenance strategies for railway tracks rely more and more on data acquired with low-cost sensors installed on in-service trains. This quasi-continuous condition monitoring produces huge amounts of data, which require appropriate processing strategies. Deep learning has become a promising tool in analyzing large volumes of sensory data. In this work, we demonstrate the potential of artificial intelligence to analyze railway track defects. We combine traditional signal processing methods with deep convolutional autoencoders and clustering algorithms to find anomalies and their patterns. The methods are applied to real world data gathered with a multi-sensor prototype measurement system on a shunter locomotive operating on the industrial railway network of the inland harbor of Braunschweig (Germany). This work shows that deep learning methods can be applied to find patterns in railway track irregularities and opens a wide area of further improvements and developments.

Keywords: Defect Detection · Deep Learning · Convolutional Autoencoder · Clustering.

1 Motivation and Introduction

The development of low-cost monitoring systems for condition based and predictive maintenance has gained increasing importance in recent years. In the railway sector, the use of in-service trains to monitor the track is considered a promising tool to collect data necessary for now- and forecasting of the track health status [13]. In this context, axle box acceleration (ABA) sensors play an important role and many different studies have shown promising results (see,

^{*} The work on signal processing presented in this paper has received funding from the European Union’s Horizon 2020 research and innovation programme and from the European Global Navigation Satellite Systems Agency under grant agreement #776402

for example, [7, 11] and references therein). It has been shown that track defects can be identified based on spectral analysis and time-frequency representations of ABA data [11]. Molodova et al. [8] proposed the use of the scale-averaged wavelet power for the automatic detection of squats. Baasch et al. [1] described a time-frequency signal separation algorithm to detect singular track defects. Furthermore, by means of time-frequency representations, the ABA data are transformed into 2D images (Figure 2a). This means that defect clustering can be framed as an image clustering problem. This fact motivates the use of deep neuronal networks in this study. Deep learning is often used for supervised learning tasks (e.g. classification) that rely on massive amounts of labeled data. In the case of railway track inspection, especially for small to mid-size infrastructure operators, the problem is that labeled data in sufficient quantity and quality are rarely available. Therefore, in this paper, we examine how deep learning methods can be applied to analyze railway track irregularities without the use of human labeled data. The idea is that once clusters in a set of identified track irregularities are found, only representative examples of each cluster need to be manually inspected by the asset manager. At best, the results of this inspection can be generalized for all members in a cluster. This would drastically reduce the visual inspection effort.

Our approach to analyze the data is the following: First, signal pre-processing is applied such that the input is well-shaped for a convolutional autoencoder (CNN-AE). Then, this CNN-AE is trained on our data set. In this way, the autoencoder learns a lower-dimensional representation of the data which can be decoded to an output similar to the input. The representations of the input data are used for further data analysis. At this point, we apply a clustering method to find similarities in the data. We further examine how outlier detection prior to the application of the CNN-AE affects the results of dimensionality reduction and clustering.

In Section 2, we describe the experimental set-up for our approach and explain the concept of autoencoders and the used clustering algorithm in more detail. Next, we evaluate and interpret the results in Section 3. Finally, Section 4 concludes and gives an outlook for further research.

2 Experimental Design

In this section, we explain all steps from acquiring the data to detecting railway track irregularities. After real-world data collection, the pre-processing steps and further data analysis are done with Python. The utilized deep learning framework is Keras which uses the Tensorflow 2.0 backend, [2].

2.1 Data Acquisition

The data used in this study were acquired with a low-cost multi-sensor prototype system developed at the German Aerospace Center (DLR, [5]). It is installed on a shunter locomotive (Figure 1) operating at the Braunschweig harbor industrial

railway network in Germany, which has a total track length of 15 km and a connection to the national mainline railway network. The ABA data are measured with a triaxial accelerometer at one of the axle boxes. The ABA sensor measures vibrations in a range of 0.8 to 8,000 kHz at a sampling rate of 20,625 Hz. A low-cost global navigation satellite system (GNSS) receiver allowing GNSS raw data acquisition and a multi-band antenna that also covers EGNSS frequency bands is used for positioning and timing. Additionally, an inertial measurement unit (IMU) at the car body (above suspension) measures accelerations and turn rates with 100 Hz. GNSS and IMU data together can be fused with a digital map in an offline process to assign locations (track ID and distance on track) and train speed to the ABA data [10]. The data analyzed here come from 60 consecutive train journeys with lengths between 12 and 372 seconds acquired in April 2016.



Fig. 1: DLR prototype of multi-sensor-measurement system (right) installed on a shunter locomotive (left) with triaxial accelerometer at the front right axle box (middle), taken from [5].

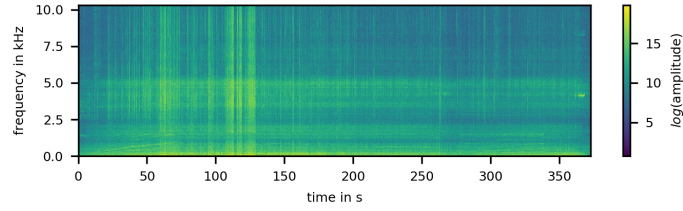
2.2 Signal Processing

The aim of the signal processing is to reduce unwanted noise in the data and to detect relevant track irregularities that can be further analyzed. The pre-processing methodologies presented here are mainly based on mathematical transformations in the field of Fourier Analysis. The outlier detection uses simple thresholding.

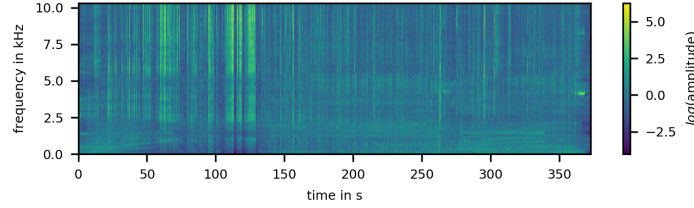
Pre-processing The measured ABA data are preprocessed as follows.

1. Time-frequency representation: In order to analyze the frequency content of the data, a short-time Fourier transform (STFT) with a window length of 2000 samples (approx. 0.1 s) and 1000 samples overlap is performed. From the complex STFT the magnitude is taken and logarithmized (Figure 2a).
2. Reduction of periodic noise: Imperfections of the wheel produce periodic impacts that lead, together with other rotating parts, to periodic noise. This noise can be removed in the cepstral domain by calculating the real Cepstrum from the STFT amplitude spectrums and applying a low-pass filter (also called lifter).

3. Removal of natural frequencies of rail and wheel: The frequency content of the ABA data is dominated by the natural frequencies of the rail and the wheel excited by the unevenness of the wheel-rail contact patch. This results in prominent horizontal bands in the time-frequency representation (Figure 2a). Those bands can be modeled by time-variant scaling of the frequency response of the dynamic wheel-rail interaction. By assuming that the frequency response is linear and the wheel-rail roughness is white, the spectrum of the system response equals the spectrum of the ABA time series multiplied by a constant. This time variant scaler is found through linear regression of the local spectrum at each time window and the average of all spectra in the time-frequency representation. The scaler is also used later on as a feature to detect outliers indicating track irregularities. Once the frequency bands are modeled, they can be simply subtracted from the time-frequency representation. The remaining signals can then be attributed to local track irregularities (Figure 2b).



(a) Time-frequency representation of raw ABA data.



(b) Time-frequency representation after pre-processing.

Fig. 2: Time-frequency representation before (a) and after (b) pre-processing

Outlier detection The outlier detection aims to find time-windows in the time-frequency representation that contain track irregularities. Those irregularities excite strong vibrations of the wheel and rail. The time-variant scaler calculated in step 3 of the pre-processing sequence is a measure of the strength of these vibrations and can therefore be used as a feature for the outlier detection. Here, an outlier is defined as a scaler with an amplitude that is higher than the average amplitude plus two times the standard deviation of all scalars.

2.3 Dimensionality reduction

In this work, we use autoencoders for the dimensionality reduction. This kind of neural networks was first introduced in [6]. The aim of an autoencoder is to learn the identity map between inputs and outputs with a specific neural network. Hence, autoencoders are a type of unsupervised learning methods as the desired output is given by the input data. An autoencoder consists of an *encoder* and a *decoder*. While the encoder maps the input data to a lower-dimensional or latent representation also named *code*, the decoder reconstructs this representation back to the original data. The smallest part of an autoencoder is typically the last layer of the encoder and the input layer of the decoder, also known as the *bottleneck*. The word *bottleneck* is also used for the latent space which contains the code. Figure 3 shows the schematic structure of a typical autoencoder.

There are a lot of variations and applications of autoencoders. Besides dimensionality reduction, autoencoders can be used for anomaly detection, image denoising, and much more. For a more detailed overview, see, for example, [4].

The input to the dimensionality reduction of our data set is the data after signal processing, namely the spectra of the processed time-frequency representations for each ABA component of all 60 journeys. Thus, each input sample consists of one frame of the time-frequency representation with the size of $1 \times 1000 \times 3$.

The encoder of this CNN-AE consists of six 1D convolutional layers and five 1D max-pooling layers in an alternating order. The decoder has six 1D convolutional layers and five 1D upsampling layers nearly symmetrical to the encoder layers. The dimension reduction only happens in the max-pooling layers. The bottleneck has an output size of 16×1 , i.e., the lower dimensional representation of the input data has dimension 16. We chose this extreme dimensionality reduction to force the model to only learn the most important features. The network consists of 1D layers because we want to examine each time frame of the time-frequency representation independently. This is due to the fact that we only investigate short track defects here. For longer and repeating irregularities, several frames could be regarded together using 2D layers in future experiments. The specific architecture can be found in Figure 4.

As the input data is not further scaled to not loose amplitude variations, a linear activation function is chosen for the first and last convolutional layer. A sigmoid activation function in the bottleneck ensures that the code is contained in the 16-dimensional unit cube of the latent space to examine the single feature vectors later. All other activation functions are tanh to include nonlinearities. The number of filters of the convolutional layers is specified in Figure 4. For all convolutional layers except for the first and last convolutional layer, the kernel size is 3 and padding is *same*. The first and last convolutional layer have kernel size 489 and 25, respectively, and both have *valid* padding. The max-pooling layers always have a pool-size of 2. All upsampling layers except the last one enlarge the data by doubling the entries. The last upsampling layer enlarges the dimension by 4.

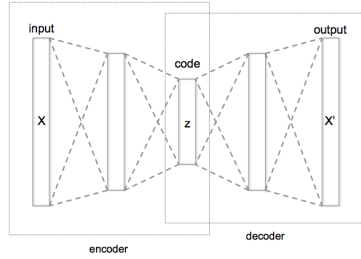


Fig. 3: The structure of a typical autoencoder.

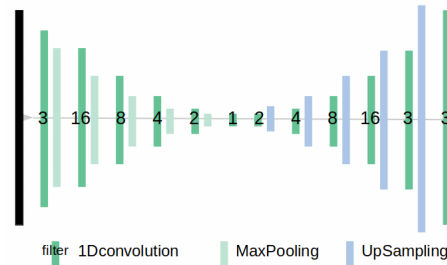


Fig. 4: Network architecture of the utilized CNN-AE.

Since the goal of this experiment is to cluster track irregularities, it was tested, if the clustering provides better results, when the outliers are extracted before the features are learned and clustered. Therefore, we did two separate experiments concerning the training. In the first experiment, the training of the autoencoder was done on the whole data set with 143420 samples and with 200 epochs. In the second experiment, only identified clusters were used in training. The outlier detection is explained in the last paragraph of Subsection 2.2. This reduced data set consists of 7642 samples, on which the training was done for 200 epochs. In both training experiments, the loss function is the mean-squared error function as it is a typical loss function for autoencoders. The utilized minimization algorithm is the ADAM algorithm with default parameters from Keras, i.e., learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-7}$.

2.4 Clustering

For clustering tasks, one can choose algorithms from a large pool of cluster algorithms. Many of them are available in certain libraries: For example, the scikit-learn library [9] contains many different cluster algorithms. A popular approach is to find a probability distribution of a Gaussian mixture model, i.e., a finite weighted sum of multidimensional Gaussian distributions, from which the given data set is most likely sampled. Each Gaussian distribution is defined by a mean and a covariance matrix, which can typically be determined by an expectation-maximization algorithm [3]. The expectation-maximization algorithm alternates between two steps, the expectation and the maximization step. Initially, the means of the single Gaussian distributions are set randomly or by a more advanced strategy. Then the expected values of the weights of the single Gaussian distributions are computed in the expectation step. These get fixed to those expected values. The maximization step determines the expected values of the parameters of the single Gaussian distributions by maximizing the probabilities of the occurrences of the data points. Then the procedure repeats until a maximum number of iterations or a convergence threshold is reached.

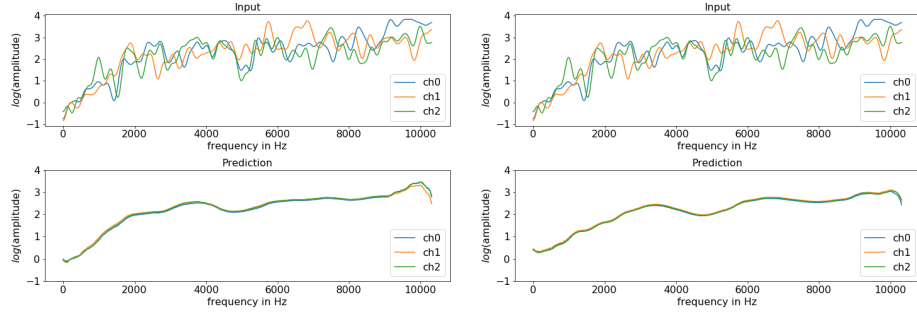
Once a Gaussian mixture model is built depending on an input set and a predefined number of components, the input samples are assigned to one cluster: For each sample, the probabilities for each mixture component are computed. The component with the highest probability defines the cluster for the sample.

In our experiments, we use Gaussian mixture models to find suitable clusters in the set of 16-dimensional representations of the input data which are obtained by the trained encoder. For the analysis, we test different numbers of mixture components and evaluate the Bayesian information criterion (BIC) [12] to examine which number of components/clusters suits best. The Bayesian information criterion depends on the likelihood function of the Gaussian mixture model, the numbers of estimated parameters k and the sample size n :

$$BIC = k \ln(n) - 2 \ln(\hat{L}).$$

Here, \hat{L} is the maximized value of the likelihood function of the Gaussian mixture model. Note that for each component, the 16-dimensional mean and a 16×16 -dimensional covariance matrix have to be computed. In general, a lower BIC is preferred.

3 Results and Interpretation



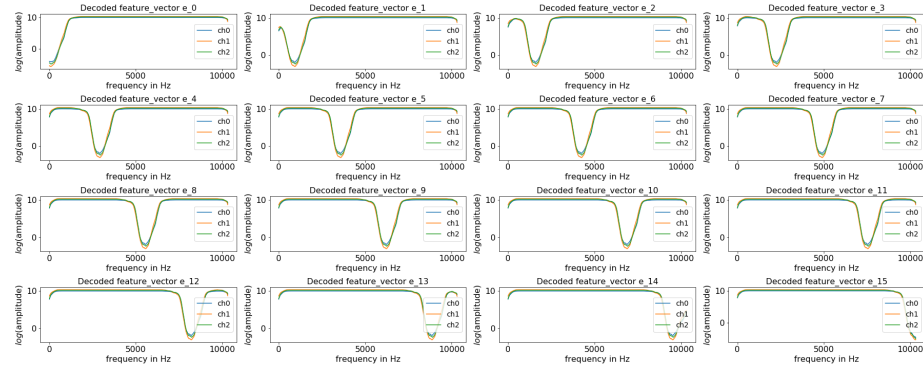
(a) Prediction after training of **AE_full**, mean squared error of 0.2054 (0.1866, 0.2012, 0.2284). (b) Prediction after training of **AE_outlier**, mean squared error of 0.2450 (0.2529, 0.2382, 0.2437).

Fig. 5: Comparison between input data and predictions of one sample (three channels). The mean squared errors of all three channels (and each single channel) are given in the sub-captions.

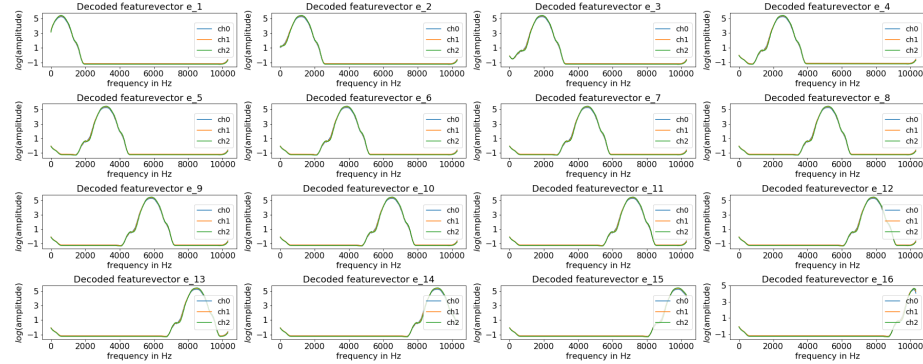
Convolutional autoencoder The training of the CNN-AE was done for 200 epochs on the full data set and on a fraction of the data set, i.e., the outliers (see Subsection 2.2), respectively. The autoencoder trained on the full data set is named **AE_full** and the one trained on the outlier set is named **AE_outlier**. In both training experiments, the losses showed a converging behaviour. For

AE_full, the overall loss was 0.1906 and the validation loss 0.1859. The training of **AE_outlier** on the outlier data set with only outlier samples delivered an overall loss of 0.3733 and a validation loss of 0.3850. These higher losses compared to the losses of **AE_full** are caused by the general higher amplitudes and more diversity within the outlier samples. Thus, it is harder for the model **AE_outlier** to reconstruct a sample.

From an analytical point of view, it is also interesting how well the input data are reproduced by the autoencoders. As the code produced by the bottleneck is very low-dimensional in comparison to the input data, we can expect that the reconstruction will be smoother resulting in further noise reduction. This is confirmed by the example in Figure 5.



(a) Decoded unit vectors after training of **AE_full**.



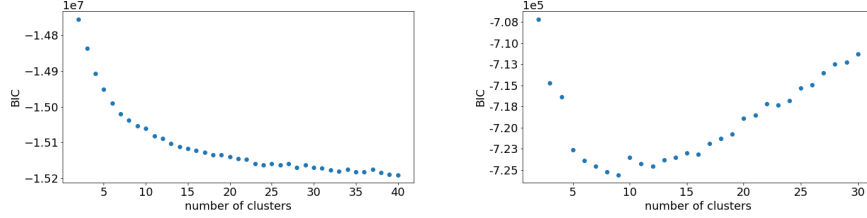
(b) Decoded unit vectors after training of **AE_outlier**.

Fig. 6: Decoded unit vectors of the 16-dimensional latent space.

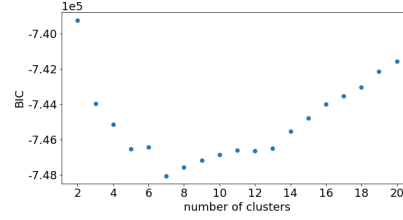
In order to examine the bottleneck in more detail, the unit vectors of the 16-dimensional latent space are decoded with the trained decoder. For both autoencoders, we obtain that each dimension of the bottleneck explores a certain range of frequencies which is visualized in Figure 6. In this way, the bottleneck act as a band-pass or band-stop filter.

Clustering Given the two trained autoencoders **AE_full** and **AE_outlier**, we cluster the code of the input data by finding a Gaussian mixture model with a predefined number of components. By varying this number and evaluating the BIC, it is possible to infer a suitable number of components.

For the autoencoder **AE_full**, the number of components is varied from 2 to 40. The resulting BIC curve is shown in Figure 7a. There, it can be seen that the BIC is more or less decreasing by increasing number of components. That suggests that the input data and its code do not build a certain cluster structure with small number of clusters.



(a) BIC for **AE_full**, code of the full data set as input set. (b) BIC for **AE_outlier**, code of the outlier data set as input set.

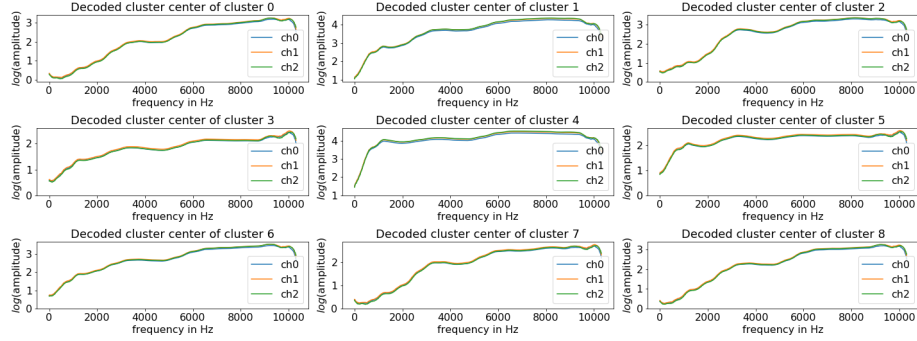


(c) BIC for **AE_full**, code of the outlier data set as input set.

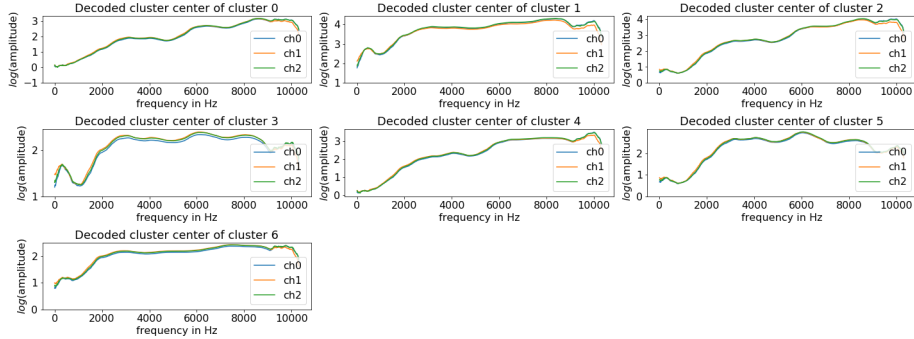
Fig. 7: Results of computing Gaussian mixture models.

In Figure 7b, we visualize the BIC for Gaussian mixture models with 2 to 30 components computed from the code of the outlier data set obtained from the autoencoder **AE_outlier**. From that, it follows that 9 components are a suitable choice. To compare both trained autoencoders, we compute Gaussian mixture models for the codes on the outlier data set obtained by the encoder of **AE_full** additionally. Figure 7c reports the BIC curve for that setting. The smallest BIC was obtained for 7 components. For the settings, where the number of components is small and the BIC is smallest, we further examine the corresponding Gaussian mixture models: To visualize the centers of the obtained clusters, the means of the best Gaussian mixture models are decoded by the decoder of the respective autoencoder (Figure 8). The differences in those decoded clusters are slight and mainly noticeable in the overall amplitude and the slope of the amplitude from lower to higher frequencies. In general, a broad spectrum with steep amplitude slope at low frequencies corresponds to a short wavelet in the time domain that could indicate a short isolated track irregularity. In contrast, a

smoother slope could indicate an increase in rail roughness. It is important to mention that this interpretation needs to be verified by means of ground truth data. The clusters built on the code obtained by `AE_full` show higher diversity due to the higher variation in the training set (Figure 8b).



(a) Decoded centers of GMM with 9 components built on code obtained by `AE_outlier`.



(b) Decoded centers of GMM with 7 components built on code obtained by `AE_full`.

Fig. 8: Decoded centers of Gaussian mixture model built on code of outlier data set obtained by different autoencoders.

Plotting cluster assignments back to railway track By means of georeferencing, the cluster assignments of each sample can be associated with a position on the corresponding track and can hence be visualized on the railway network map (Figure 9). This information can be used to guide specific maintenance actions. Here, we use the cluster assignments of the outlier data set obtained by the Gaussian mixture model with 9 components, which was computed from the codes of the encoder of the trained model `AE_outlier`. It can be seen that some clusters accumulate at certain sections of the network. In the north-west of the network cluster 1 is predominant. In this area, different scarp and coal loading sites are situated. The loading leads to dirt on the track that increases the track roughness and can also lead to track defects. It still needs to be verified whether the clusters can be associated with specific track defects.

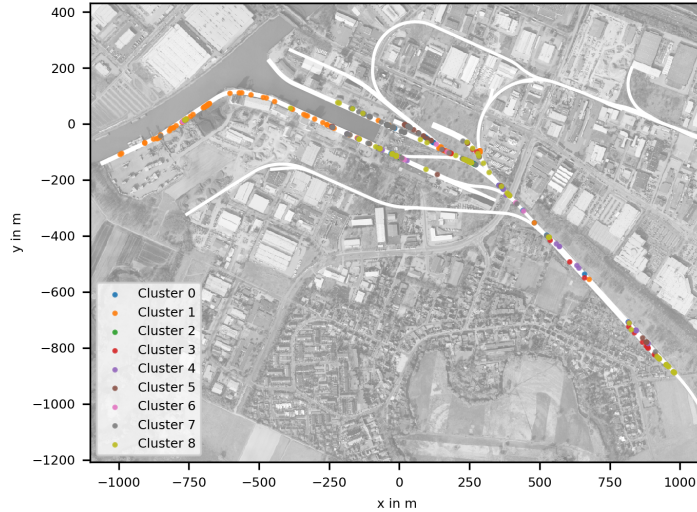


Fig. 9: Cluster assignments mapped on industrial railway network at the Braunschweig harbour.

4 Conclusion and Outlook

In this paper, we presented an approach to analyze real world data, i.e., ABA data from train journeys at an industrial railway network, with signal pre-processing and unsupervised machine learning methods to track irregularities on railway tracks.

Our chosen unsupervised machine learning approach based on deep learning, namely convolutional autoencoders, and Gaussian mixture models is capable of clustering ABA anomalies. The clustering algorithm was applied to the full data set and to outliers detected with signal processing methods only. Clustering the full data set did not lead to reliable results. In contrast, when only outliers were considered, an optimal number of clusters was found. Nevertheless, the differences between the clusters were small, which made the interpretation difficult.

Therefore, validation with ground truth data is important and should be considered in the future. If the clusters match existing failure modes, this would also motivate to adapt the presented CNN-AE methodology towards a supervised classification approach.

Furthermore, it could be tested if the clustering could be improved by integrating the clustering in the deep learning architecture, i.e., using more advanced techniques such as deep embedded clustering like, for example, in [14, 15].

In this sense, one prospective aim is to unite all steps after the data acquisition, namely the pre-processing including denoising and a first outlier detection, the dimensionality reduction and the clustering, to one deep learning model. For this, a suitable architecture and loss function have to be found.

For further research, it is also of interest to examine more consecutive time frames together to track longer irregularities. From the methodical point of view, anomaly detection in sequential data is not that well studied now and offers the chance for developing new deep learning methods.

References

1. Baasch, B., Roth, M., Havrila, P., Groos, J.C.: Detecting singular track defects by time-frequency signal separation of axle-box acceleration data. In: WCCR 2019 (2019)
2. Chollet, F., et al.: Keras. <https://keras.io> (2015)
3. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* **39**(1), 1–22 (1977)
4. Dong, G., Liao, G., Liu, H., Kuang, G.: A review of the autoencoder and its variants: A comparative perspective from target recognition in synthetic-aperture radar images. *IEEE Geoscience and Remote Sensing Magazine* **6**(3), 44–68 (2018)
5. Groos, J.C., Havrila, P., Schubert, L.: In-service railway track condition monitoring by analysis of axle box accelerations for small to mid-size infrastructure operators. In: The British Institute of Non-Destructive Testing (ed.) *Proceedings of WCCM 2017 congress* (2017)
6. Hinton, G.E., Zemel, R.S.: Autoencoders, minimum description length and helmholtz free energy. In: *Advances in neural information processing systems*. pp. 3–10 (1994)
7. Li, Z., Molodova, M., Núñez, A., Dollevoet, R.: Improvements in axle box acceleration measurements for the detection of light squats in railway infrastructure. *IEEE Transactions on Industrial Electronics* **62**(7), 4385–4397 (2015)
8. Molodova, M., Li, Z., Núñez, A., Dollevoet, R.: Automatic detection of squats in railway infrastructure. *IEEE Transactions on Intelligent Transportation Systems* **15**(5), 1980–1990 (2014)
9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011)
10. Roth, M., Baasch, B., Havrila, P., Groos, J.C.: Map-supported positioning enables in-service condition monitoring of railway tracks. In: *International Conference on Information Fusion (FUSION)*. pp. 2346–2353 (2018)
11. Salvador, P., Naranjo, V., Insa, R., Teixeira, P.: Axlebox accelerations: Their acquisition and time-frequency characterisation for railway track monitoring purposes. *Measurement* **82**, 301–312 (2016)
12. Schwarz, G., et al.: Estimating the dimension of a model. *The annals of statistics* **6**(2), 461–464 (1978)
13. Weston, P.F., Roberts, C., Goodman, C.J., Ling, C.S.: Condition monitoring of railway track using in-service trains. In: *2006 IET International Conference On Railway Condition Monitoring*. pp. 26–31 (2006)
14. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: *International conference on machine learning*. pp. 478–487 (2016)
15. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 5147–5156 (2016)